COBOS: Cooperative Backoff Adaptive Scheme for Multirobot Task Allocation

Cheng-Heng Fua, *Student Member, IEEE,* and Shuzhi Sam Ge, *Senior Member, IEEE*

***Abstract*—In this paper, the cooperative backoff adaptive scheme (COBOS) is proposed for task allocation amongst a team of heterogeneous robots. The COBOS operates in regions with limited communication ranges, and is robust against robot malfunctions and uncertain task specifications, with each task potentially requiring multiple robots. The portability of tasks across teams (or when team demography changes) is improved by specifying tasks using basis tasks in a matrix framework. The adaptive feature of COBOS further increases the flexibility of robot teams, allowing robots to adjust their actions based on past experience. In addition, we study the properties of COBOS: operation domain; communication requirements; computational complexity; and solution quality; and compare the scheme with other task-allocation mechanisms. Realistic simulations are carried out to verify the effectiveness of the proposed scheme.**

***Index Terms*—Cooperative backing off, disjoint networks, fault tolerance, multirobot tasks, task allocation.**

I. Introduction

A NUMBER of schemes have been proposed for fault-tolerant multirobot task allocation in ST-SR-IA[1] domains. A well-known architecture is ALLIANCE [2] (and extended with parameter adaptivity in L-ALLIANCE [3]), which integrates *impatience* and *acquiescence* into each robot. Another approach is the broadcast of local eligibility (BLE) technique [4] that uses cross inhibition of behaviors between robots in a team. A task-assignment architecture that uses task templates and the prioritization of task instances in a task-assignment planner has been proposed [5] for transportation applications in unknown, but static, environments. The M+ protocol [6] has a task-allocation layer, with the negotiation process based on the contract net protocol.

Auctioning schemes using explicit communications have been used in various forms in allocation schemes. Robust multirobot cooperation may be achieved through the use of market-based approaches [7], [8]. A dynamic role-assignment scheme that represents roles as control modes in hybrid automatons has also been proposed [9]. MURDOCH [10] uses an auctioning approach with a publish/subscribe communication model. A role-assignment strategy based on multithreaded computer programming was used to resolve any risks and conflicts that may arise during dynamic role swapping [11]. In cases where communication losses are considered (e.g., in MURDOCH [10]), it is often assumed that persistent communication loss implies robot failure, when in fact the robot may still be performing the task adequately. For instance, for reconnaissance missions, due to security and practical issues, communication ranges for robots may be intentionally reduced. Thus, in the absence of team-wide communications, robots should be able to respond appropriately.

TABLE I
NOMENCLATURE

x_k	the k-th element of a vector $\mathbf{X} \in \mathbb{R}^n$;
$\|X\|$	the number of elements in a set X;
r_j	the robot j;
n_j	the number of robots r_j has communication links with;
$\mathbf{TSM}(j)$	the Task Utility Matrix compiled by r_j;
$\mathbf{TSM}_{ij}(j)$	the (i,j)-th element of $\mathbf{TSM}(j)$;
$\mathbf{TSM}^{-}(j)$	a sub-matrix made up of selected rows and columns of $\mathbf{TSM}(j)$;
$\mathcal{B}$, $\mathbf{B}$	the set, or vector form, of Basis Tasks;
$b_i(\ell_i)$	$(b_i \in \mathcal{B})$ the i-th basis task with the list of arguments it accepts, ℓ_i;
n_b	the number of basis tasks in $\mathcal{B}$;
$\mathbf{L}$	the arguments accepted by basis tasks in $\mathbf{B}$;
$\mathcal{T}_i$	the i-th macro task;
$\mathcal{L}_i$	the physical location associated with $\mathcal{T}_i$;
ϕ_i	the number of robots required by $\mathcal{T}_i$;
$E_i(t)$	the expected maximum duration of $\mathcal{T}_i$;
$\mathcal{T}_{i,k}$	the k-th sub-macro task of $\mathcal{T}_i$;
$\mathbf{T}_i$	the Task Specification Matrix of $\mathcal{T}_i$;
$T_{i,dp}$	the set of tasks that must be completed before $\mathcal{T}_i$ can start;
$\|\mathcal{T}_i\|$	the number of sub-macro tasks in $\mathcal{T}_i$;
$\|\mathcal{T}_{i,j}\|$	the number of basis tasks in $\mathcal{T}_{i,j}$;
T^p	the ordered set of high priority tasks;
T^{atv}	the ordered set of active, but not high priority tasks;
T^{ach}	the ordered set of archived tasks;
T_{ls}	the ordered set of all the tasks given to the robots;
n_{ls}	the number of tasks given to the robots;
n_{atv}	the number of tasks in a subnetwork that can be considered, given the number of robots in that subnetwork, and the total number of robots required for each of these tasks;
$\mathbf{A}_j$	the vector containing the utility level r_j has with each of the n_b basis tasks;
$\mathcal{T}_{rj}$	the task r_j is currently performing;
W	the physical workspace of the team;
N	the set of disjoint subspaces in W;
n_{sn}	the number of disjoint networks in W;
$\mathbf{S}_j$	$(\mathbb{Z}^{n_b \times 1})$ the Task Success Matrix of r_j.

Manuscript received February 1, 2005; revised April 29, 2005. This paper was recommended for publication by Associate Editor G. Sukhatme and Editor S. Hutchinson upon evaluation of the reviewers' comments. This paper was presented in part at the IEEE International Symposium on Intelligent Control, Taipei, Taiwan, R.O.C., September 2004.

C.-H. Fua is with the NUS Graduate School of Integrative Sciences and Engineering, National University of Singapore, Singapore 117576 (e-mail: c.fua@nus.edu.sg).

S. S. Ge is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: elegesz@nus.edu.sg).

Digital Object Identifier 10.1109/TRO.2001.855992

[1]For convenience, we use the notations proposed in [1] (ST-MR-IA: single-task robots, multirobot tasks, instantaneous assignment systems; SR: single-robot tasks; TA: time-extended assignment).

For domains with MR tasks, coalitions may be formed [12]. However, with uncertain (and/or incomplete) task specifications, the evaluation of team effectiveness is impaired. Consider the task of clearing an obstructing rock detected during a mission. It is difficult for a remote user to determine, using only sensor information (e.g., video), the exact weight of the rock, and the number of robots required, which may vary across different teams. It is also beneficial to use a robot's history of failures to assess and determine *which* of its onboard capabilities is the most likely cause of failures. A number of research papers, such as [10] and [13], have considered the issue of adapting to robot malfunctions. However, fault detection based on equipment access (e.g., in [13]) may not be applicable, such as when a resource suffers from diminished capability (e.g., a worn-out gripper that reduces its ability to carry heavy loads) and the robot still has continued access to the resource.

The main contributions of this paper are: 1) a matrix-based approach proposed for specifying tasks (accounting for task dependencies and alternative methods of performing tasks) using basic task-achieving behaviors. It improves the portability of tasks across different robot teams and allows robots to make use of task success/failure histories to detect and adapt to device imperfections and malfunctions; and 2) fault-tolerant task allocation is achieved using a cooperative backoff adaptive scheme (COBOS),[2] designed to improve robot autonomy in *ST-**MR**-IA* domains with uncertain task specifications, and with disjoint communication networks.

Remark 1.1: "*Backing off*" that is used in communications (e.g., IEEE 802.11 networks) is competitive [19]. Conventional backing off cannot solve the problem considered here. For example, consider a system with one task. A robot that fails at the task will first back off, and continue attempting the task after each backoff period. The same will occur for any robot that claims and fails at the task after robot 1 backs off. This causes the robots to vie for the task, instead of cooperating. In contrast, COBOS explicitly synthesizes cooperation into the system.

II. Cooperative Backoff Adaptive Scheme (COBOS)

The following assumptions are made in this paper.

Assumption 1: A robot determines the task status of tasks handled by robots it is currently in contact with via wireless broadcasting. A task's status indicates whether a robot has started the task, or it has been completed. A robot broadcasts its current activities for this purpose. The broadcasting range of each robot is limited and finite. If a robot loses contact with another, and cannot ascertain a task's current status, the robot will keep the task status unchanged, until communication links are established again.

Assumption 2: The generation of tasks is entirely random, and no *a priori* planning is possible. This is true for robot operation in unstructured and dynamic environments where conditions fluctuate greatly, and only instantaneous assignment is possible.

[2]Deliberative cooperation is examined in this paper, and only the task-allocation layer is considered. The robots are assumed to possess lower level behaviors (e.g., [4] and [5]) to carry out their allocated tasks while satisfying task constraints. A comprehensive overview of issues in multirobot cooperation may be found in [1].

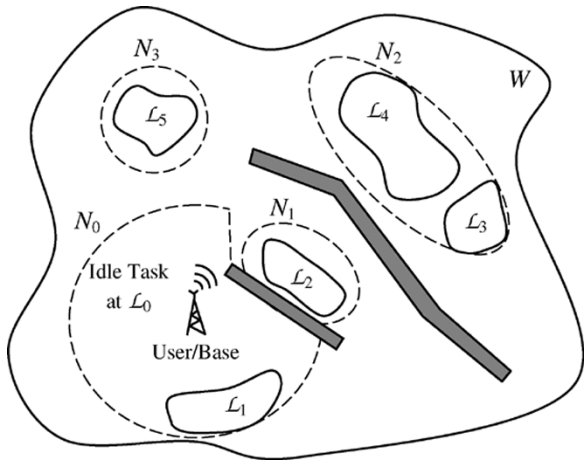


Fig. 1. Existence of subnetworks in workspace due to deep fading and signal attenuation.

A. Disjoint Broadcast Networks

In real life, communications may face *persistent losses* (e.g., due to deep fading or large interferences), causing loss of contact over extended time periods, and is the problem that COBOS is concerned with. Another form occurs when robots are within the same broadcast region and experience *temporary* packet losses. Most work dealing with communications imperfections deal with the second type, which may be solved through more frequent auctioning (e.g., in [13]) and with standard communications protocols. Furthermore, a loss of access to communications does not necessarily imply device failure. It has important implications on task allocations, and is thus treated in detail in this paper.

Let $N = \{N_{i_{\rm sn}} \mid N_{i_{\rm sn}} \in W, \text{ for } i_{\rm sn} = 0, \ldots, n_{\rm sn}\}$ be disjoint subnetworks of the workspace W, and $n_{\rm sn}$ is the number of subnetworks. Each $N_{i_{\rm sn}}$ is a network resulting from the separation of robots from the main broadcasting network. Since the subnetworks arise due to the locations of each task, these locations are related to one of the subspaces, i.e., $\mathcal{L}_i \subseteq N_{i_{\rm sn}}$, where $\mathcal{L}_i$ is the location where $\mathcal{T}_i$ is to be conducted (shown in Fig. 1). Furthermore, information regarding a new task is obtained only when a robot is in N_0, where the user is located, or another robot that has claimed the new task enters the subnetwork. For overlapping subnetworks, rebroadcasting of information by robots may be done. However, disjoint subnetworks may still exist at different locations due to reasons cited earlier, and the formulation still holds.

B. Formal Description of Tasks

The COBOS is implemented on each robot in the team. Each robot makes use of sensor information (including broadcasted information) to construct and adapt its internal models (shown in Fig. 2). The task model consists of the properties and descriptions of tasks robot r_j is currently aware of. The task description

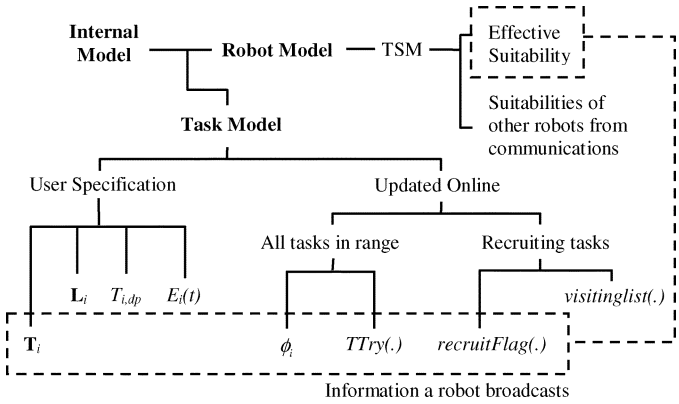


Fig. 2. Elements of each robot's internal models.

aims to make the information regarding basic task-achieving behaviors required for individual tasks readily available to robots.[3] Although similar to task abstractions [7], the proposed approach allows explicit specification of the *capabilities* a robot must possess to be eligible for a task. This would mean the decomposition of each primitive task into even smaller subtasks (capabilities), with the constraint that they must all be achievable by one robot. The proposed representation improves the handling of information regarding robot capabilities, alternative ways of performing tasks, and task dependencies.

Definition 2.1 (Basis Tasks): Basic high-level tasks (or behaviors) out of which other, more complex tasks may be composed. Basis tasks may also be seen as basic abilities, motor schemas [21], or resources [10].

Definition 2.2 (Macro Tasks): Complex tasks that are composed from basis tasks. These will be referred to as "tasks" for the remainder of the paper.

A task $\mathcal{T}_i$ is defined as a quadruplet $(\mathbf{T}_i, \mathbf{L}_i, T_{i,dp}, E_i(t))$, and the individual elements are described as follows.

1) Operator $\odot$: Let $\mathcal{B}$ be the known set of n_b basis tasks, each accepting its own list of arguments ℓ (which may be sets of coordinates, areas to explore, etc.), available for task specification, such that $b_1(\ell_1), b_2(\ell_2), \ldots, b_{n_b}(\ell_{n_b}) \in \mathcal{B}$. It may not reflect *only* the capabilities of current members of the team, and can encompass a larger set of basis tasks. The basis tasks and the corresponding list of arguments is cast in vector form as $\mathbf{B} = [b_1, b_2, \ldots, b_{n_b}]^T$ and $\mathbf{L} = [\ell_1, \ell_2, \ldots, \ell_{n_b}]^T$, respectively. A task $\mathcal{T}_i$ is composed of a combination of basis tasks, i.e., $\mathcal{T}_i \subset \mathcal{B}$, and is written as

$$\begin{aligned}\mathcal{T}_i &= \mathbf{T}_i \odot \mathbf{L} \vdash \mathbf{B} \\ &= \{b_k(\ell_k) : \text{ for } k = 1, \ldots, n_b \text{ and } \mathbf{T}_i(k) \neq 0\} \end{aligned} \quad (1)$$

with $\mathbf{T}_i \in \mathbb{R}^{1 \times n_b}$ and its kth element defined as

$$\mathbf{T}_i(k) = \begin{cases} 1, & \text{if } \mathcal{T}_i \text{ requires } b_k \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Equation (1) may be interpreted as $\mathcal{T}_i$, under $(\vdash)$ $\mathbf{B}$, requires all the basis tasks b_j for which $\mathbf{T}_i(j) = 1$, and accepts ℓ_j as arguments. For simplicity, $\vdash \mathbf{B}$ will be omitted from task specifications in this paper. The operator $\odot$ acts like the matrix multiplication, except that the result is a *set* of the required basis tasks. This facilitates the manipulation and use of task information through normal matrix operations.

The nature of a task refers to the generic type of task that it belongs to, and the related abilities (perhaps in varying degrees) required of robots. For instance, two tasks, both involving grasping and lifting an object (of different weights), are of the same nature, i.e., both are picking-type tasks. Tasks are decomposed into two main parts, the nature and the details of the task, characterized by $\mathbf{T}$ and $\mathbf{L}$, respectively. A more general task is stated as

$$\begin{aligned}\mathcal{T}_i &= \{\mathcal{T}_{i,1} | \mathcal{T}_{i,2} | \ldots | \mathcal{T}_{i,n}\} \,\&(T_{i,dp}) \\ &= \{(\mathbf{T}_{i,1} \odot \mathbf{L}_{i,1}) | \ldots | (\mathbf{T}_{i,n} \odot \mathbf{L}_{i,n})\} \,\&(T_{i,dp}) \\ &= \left(\begin{bmatrix} \mathbf{T}_{i,1} \\ \vdots \\ \mathbf{T}_{i,n} \end{bmatrix} \odot [\, \mathbf{L}_{i,1} \quad \ldots \quad \mathbf{L}_{i,n} \,] \right) \&(T_{i,dp}) \\ &= (\mathbf{T}_i \odot \mathbf{L}_i) \,\&(T_{i,dp}) \end{aligned} \quad (3)$$

where

$$\begin{aligned}&\mathcal{T}_{i,k} = \mathbf{T}_{i,k} \odot \mathbf{L}_{i,k}, \quad \text{for } k = 1, 2, \ldots, n \\ &\mathcal{T}_{i,j} \not\subseteq \mathcal{T}_{i,k}, \quad \text{for } j, k = 1, 2, \ldots, n \text{ and } j \neq k \end{aligned} \quad (4)$$

and $T_{i,dp}$ is the set of tasks that must be performed before $\mathcal{T}_i$ can be activated, and is indicated by $\&(\cdot)$. $\odot$ operates on the $n \times m$ and $m \times n$ matrices to form an $n \times 1$ vector, and the kth element of $\mathbf{T}_i \odot \mathbf{L}_i$ in (3) is given by $\mathbf{T}_{i,k} \odot \mathbf{L}_{i,k}$. The sign $|$ means that any robot possessing all the capabilities specified in any of the sets, $\mathbf{T}_{i,k}$ (referred to as *submacro tasks*), will be able to handle $\mathcal{T}_i$.

Consider a mission that requires a gathering task to bring objects to a point A, and the transportation of these objects from point A to B across a river. These two tasks may be specified as

$$\begin{aligned}\mathcal{T}_1 &= \{\text{search(Area)}, Tpt\text{\_land(current, A)}\} \\ &= [\,1 \quad 1 \quad 0 \quad 0\,] \odot \begin{bmatrix} \{\text{Area}\} \\ \{\text{current, A}\} \\ \times \\ \times \end{bmatrix} \vdash \mathbf{B} \\ \mathcal{T}_2 &= \{\mathcal{T}_{2,1} | \mathcal{T}_{2,2}\} \& (\mathcal{T}_1) \\ &= \left(\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \odot \begin{bmatrix} \times & \times \\ \times & \times \\ \{A, B\} & \times \\ \times & \{A, B\} \end{bmatrix} \right) \& (\mathcal{T}_1) \end{aligned}$$

with

$$\begin{aligned}\mathbf{B} &= [\text{search(Area)}, Tpt\text{\_land}(a, b), \\ &\qquad Tpt\text{\_water\_surface}(a, b), Tpt\text{\_Flight}(a, b)]^T \\ \mathcal{T}_{2,1} &= \{Tpt\text{\_water\_surface(A, B)}\} \\ \mathcal{T}_{2,2} &= \{Tpt\text{\_Flight(A, B)}\} \end{aligned}$$

where $\times$ in the $\mathbf{L}$ matrices is not considered. $\mathcal{T}_2$ may be completed by a robot that is able to transport a load either across water or through air, but it is not necessary for a robot to have both capabilities.

[3]There are several ontological issues involved in task specification and combining basis tasks. However, for our purposes here, we are mainly interested in the problem of allocating a set of tasks (already specified) to a team of robots. As such, what is proposed here acts to constrain the type of information required, and is a general structure that users may follow when specifying tasks for the robot team.

2) Expected Maximum Task Durations: The expected maximum duration of a task, $E_i(t)$, is specified by the user. When the task is not completed after $E_i(t)$, it means that either the subteam assigned to the task may have encountered problems and have possibly failed, or none of the robots are doing the task. Such tasks will be referred to as *lapsed tasks*. This triggers a reallocation, and either replaces or enlarges the current subteam. The deadline will then be reset, and the new team will be given $E_i(t)$ to complete the task.

C. Task Suitability Matrices (TSM)

The set of tasks a robot is aware of may be partitioned into three ordered sets: 1) priority tasks T^p; 2) active tasks T^{atv}; 3) archived tasks T^{ach}. Tasks with precedence requirements are put in T^{ach}, while the preceding tasks are included in T^{atv}. T^p consists of high-priority tasks. Any new task that a robot receives during the mission will be added, according to its priority, into one of the task sets. The tasks may be written as $T_{ls} = \{T^p, T^{atv}, T^{ach}\}$ with a total of n_{ls} tasks, which may be larger than the total number of robots in the system.

The task suitability matrix (TSM), i.e., the team model, reflects the suitability levels each robot has for each task in T_{ls}. Task suitability is defined as follows.

Definition 2.3 (Task Suitability): The suitability of a robot for performing a certain task, based on the set of intrinsic capabilities it possesses and the abilities required for successful execution of that task. The computation of suitability levels will be detailed in the subsequent parts of this section.

Remark 2.1: Task suitability is slightly different from how utility is normally used. Instead of predicting the expected solution quality a robot-to-task allocation yields, suitability reflects how qualified a robot is in performing a certain task, based on the capabilities it possesses.

Each robot in the team maintains its own TSM (of size $n_j \times n_{ls}$), where n_j is the current number of robots that r_j detects (based on communications). The number of robots in the system is not assumed to be fixed. Thus, a robot expands its TSM when it detects new robots. The suitability of r_j for each basis task in $\mathcal{B}$ may be specified in advance as $\mathbf{A}_j = [a_{j1}, \ldots, a_{jn_b}]^T$, with $a_{jk} \in [0,1]$ for $k = 1, 2, \ldots, n_b$. The suitability of robot r_j with $\mathcal{T}_i$ is an average of the suitabilities the robot has with the basis tasks needed by $\mathcal{T}_i$. If a task has more than one submacro task, suitability is given by the maximum suitability the robot has with the submacro tasks

$$\text{TSM}_{ji} = \begin{cases} \max\limits_{\gamma=1,\ldots,|\mathcal{T}_i|}(x_{ji,\gamma}), & \mathbf{X}_{ji} \neq \varnothing \\ -\infty, & \text{otherwise} \end{cases} \tag{5}$$

where $x_{ji,\gamma}$ is the γth element of $\mathbf{X}_{ji} \in \mathbb{R}^{|\mathcal{T}_i|}$, and reflects the suitability of r_j for $\mathcal{T}_{i,\gamma}$. The matrix $\mathbf{X}_{ji}$ is given by

$$x_{ji,\gamma} = \begin{cases} \dfrac{1}{|\mathbf{T}_{i,\gamma}|}\mathbf{T}_{i,\gamma}\Upsilon_j\mathbf{A}_j, & \text{if for } k = 1, \ldots, n_b, \\ & \mu_{j,k}a_{jk} \neq 0 \quad \forall \mathbf{T}_{i,\gamma}(k) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $\Upsilon_j = \text{diag}(\mu_{j,1}, \mu_{j,2}, \ldots, \mu_{j,n_b})$ and $\mu_{j,k} \in [0,1]$ for $k = 1, 2, \ldots, n_b$. The values of $\mu_{j,k}$ are initialized to 1 and adapted during runtime. The combined matrix $\Upsilon_j\mathbf{A}_j \in \mathbb{R}^{n_b \times 1}$ is the *effective suitability* of r_j for the basis tasks in $\mathcal{B}$. Each robot will broadcast its suitability for each of the tasks, and also updates its $\mathbf{TSM}(j)$ according to received values. It also broadcasts the task that it chooses to perform, $\mathcal{T}_{rj}$, together with any related information (such as $\mathbf{T}_i$). If there are more tasks than available robots, higher priority tasks will be serviced first by using a submatrix of the TSM.

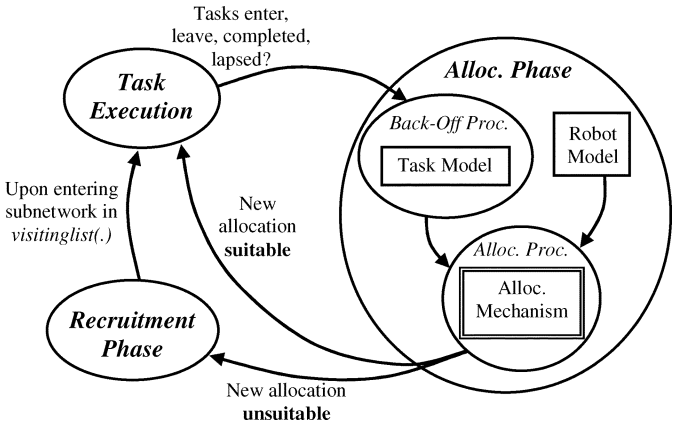


Fig. 3. Components and phases of the COBOS.

Remark 2.2: Suitability is calculated based on the compatibility of a robot's *intrinsic* abilities to the *nature* of the task, and involves only $\mathbf{T}$. *Extrinsic* factors, such as time and distance, may be incorporated through $\mathbf{L}$. For example, the distance of a robot to the transportation task (Section II-B) may be computed using the second argument to $Tpt\_\text{land}(a, b)$. A number of approaches (e.g., MVERT [22]) based largely on metrics such as distance have been proposed. Different utility values can be combined via a weighted summation. However, this complicates computation, since different tasks demand different kinds of calculations, and it is practically difficult to estimate factors like task-completion time to allow meaningful comparison of metrics computed by different robots, especially if task information is uncertain.

D. Fault Tolerance and Uncertain Task Specifications

COBOS consists of the allocation and the recruitment phases (Algorithms 1 and 2), and the main structure is shown in Fig. 3. The idle task $\mathcal{T}_0$ (at $\mathcal{L}_0$) may be a gathering point near the user. For a robot r_j, the lapsecounter$_j(\mathcal{T}_i)$ keeps track of the amount of time that has elapsed since it received $\mathcal{T}_i$, $TTry_j(\mathcal{T}_i)$ records the number of robots that have attempted $\mathcal{T}_i$, and $TTry_j(\mathcal{T}_k)$ is initialized to one, visitinglist$_j(\mathcal{T}_k)$ is an ordered list (in ascending priority, with $\mathcal{T}_0$ as the first in the list) of tasks that has lower priority than $\mathcal{T}_k$. The flag recruitFlag$(\mathcal{T}_i)$ indicates if a task is a recruiting task. The number of robots required for a task, ϕ_i, is initialized to one.

Robots enter the allocation phase when: 1) the tasks under consideration change (when new/recruiting tasks are detected in the subnetwork, or tasks are completed/leave the subnetwork); or 2) one or more tasks a robot is currently considering has lapsed. A robot r_j first examines the lapsed tasks it has detected, and enters the backoff process to adjust the number of robots

required for $\mathcal{T}_k$ (lines 7–14 of Algorithm 1), and adapt to uncertainties in $\mathcal{T}_k$'s specifications. If a robot is the first to fail at completing a task, it backs off by setting its suitability for that task to $-\infty$, and lets other robots attempt this task. If the first robot is unsuitable for the task, the second robot will succeed. Otherwise, it is highly probable that more robots are needed for the task, and r_j increases ϕ_k. Robots that are not allocated any tasks return to N_0 (line 34 of Algorithm 1), where lapsed tasks in all subnetworks are considered (lines 2 and 3 of Algorithm 1), and will be deployed to these tasks.

Algorithm 1 COBOS Allocation Phase
1: **if** (r_j is in N_0) **then**
2: Broadcast all detected lapsed tasks.
3: $T_j^- =$ tasks performed by robots in N_0 + recruiting tasks + all lapsed tasks detected in the broadcast.
4: **else**
5: $T_j^- =$ tasks performed by robots in r_j's subnetwork N_{j_s} + recruiting tasks.
6: Reset lapsecounter$(\cdot)$ of all recruiting tasks.
7: *// Backoff Process: Adapt parameters of lapsed tasks.*
8: **for all** (lapsed tasks $\mathcal{T}_k$ in T_j^-) **do**
9: **if** (All robots doing $\mathcal{T}_k$ has started OR no robots are doing $\mathcal{T}_k$) **then**
10: **if** ($TTry(\mathcal{T}_k) > 1$) **then**
11: ϕ_k++.
12: **if** ($\mathcal{T}_k = \mathcal{T}_{rj}$) **then**
13: r_j restore its previous suitability for $\mathcal{T}_k$.
14: **else if** ($TTry(\mathcal{T}_k) = 1$) **then**
15: Set the suitability of r_{j*} for $\mathcal{T}_k$ as $-\infty$, where $\mathcal{T}_{rj*} = \mathcal{T}_k$.
16: $TTry(\mathcal{T}_k)++$.
17: **else**
18: Set the suitability of r_{j*} for $\mathcal{T}_k$ as $-\infty$.
19: *// Compile* $\mathbf{TSM}^-(j)$
20: Let $n_{atv} = \arg\max_{y=1,\ldots,n^-}(\sum_{k=1}^{y}\phi_k \le n_j)$, where n^- is the number of tasks in T_j^-.
21: *// Allocation Process.*
22: Get possible new allocation.
23: *// Examine solution. Is recruitment needed?*
24: **for all** ($\mathcal{T}_k$ in the first n_{atv} tasks of T_j^-) **do**
25: **if** not enough suitable robots allocated to $\mathcal{T}_k$ **then**
26: For all robots assigned to $\mathcal{T}_k$: Enter $\mathbf{recruitphase}(\mathcal{T}_{\mathbf{k1}}, \mathbf{r_j})$.
27: **for all** (robots that have no tasks in the new allocation) **do**
28: **if** (there exists a $(n_{atv}+1)$th task, $\mathcal{T}_{k2}$ in T_j^-) **then**
29: Robot enters the $\mathbf{recruitphase}(\mathcal{T}_{\mathbf{k2}}, \mathbf{r_j})$.
30: **else**
31: **if** ($r_j = r_{j*}$) **then**
32: Robot remains in subnetwork.
33: **else**
34: Robot goes to (or remains in) N_0.
35: r_j broadcasts decision, and related information.

In the recruitment phase, undersized teams visit other task locations that they are aware of (which may not be the entire set of user-specified tasks), while still holding on to the task $\mathcal{T}_k$. This brings awareness of $\mathcal{T}_k$ into other subnetworks, forcing robots in these subnetworks to enter the allocation phase (lines 1–6 of Algorithm 1). This causes a limited exchange of robots, task information, and robot resources between subnetworks. When a robot finds (from its **TSM**) that the number of robots required for a task is greater than the number of robots that can perform the task, the team is deemed to be unable to perform the task, and the task is put into $\mathcal{T}^{ach}$ until the number of capable robots is enough (e.g., when additional robots enter the team).

Algorithm 2 COBOS recruitphase$(\mathcal{T}_k, r_j)$
1: *// Label $\mathcal{T}_k$ as a recruiting task.*
2: Set recruitFlag$(\mathcal{T}_k) =$ TRUE.
3: Set $\mathcal{T}_{rj} = \mathcal{T}_k$.
4: Compile visitinglist$_j(\mathcal{T}_k)$.
5: *// Move to recruitment locations in sequence.*
6: Instead of moving to $\mathcal{L}_k$: Move to the location of $\mathcal{T}_{k*}$, the first task in visitinglist$_j(\mathcal{T}_k)$.
7: **if** (visitinglist$_j(\mathcal{T}_k)$ is empty) **then**
8: Goto Idle location and wait.
9: **if** ($\mathcal{L}_{k*}$ is reached) **then**
10: Remove $\mathcal{T}_{k*}$ from visitinglist$_j(\mathcal{T}_k)$
11: Set recruitFlag$(\mathcal{T}_k) =$ FALSE.
12: Exit $\mathbf{recruitphase}(\cdot)$.

E. Adaptation of Internal Robot Model

Let the task success matrix be $\mathbf{S}_j \in \mathbb{Z}^{n_b \times 1}$. It identifies the ability of a robot (as reflected by the corresponding basis task) that is causing persistent failure in the tasks it attempts, and is updated upon the successful completion of a task, when it hands over a task to another, or when the related TSM value is restored (line 13 in Algorithm 1). The matrix is initialized to zero, and is updated by

$$\mathbf{S}_j = \mathbf{S}_j + \eta_j f\left(\mathbf{X}_{ji}^T \mathbf{T}_i\right) \tag{7}$$

where

$$\eta_j = \begin{cases} +1, & \text{Successful completion of } \mathcal{T}_i \\ -1, & \text{Failure of } \mathcal{T}_i \end{cases}$$

and $f(\cdot)$ produces a matrix of the same size as its argument, but converts all elements greater than zero into one. The values of Υ_j in (6) are calculated from $\mathbf{S}_j$ as

$$\mu_{j,k} = \begin{cases} 1, & \text{if } x_{j,k} \ge 0 \\ \frac{1}{-x_{j,k}+\epsilon}, & \text{otherwise} \end{cases} \tag{8}$$

where $k = 1, 2, \ldots, n_b$. The parameter $\epsilon > 0$ is user defined and determines the sensitivity of r_j's task suitabilities to the number of task failures it experiences. Intuitively, (8) means that the suitability of the robot for the basis task is reduced as the number of failed tasks associated with the basis task increases. Furthermore, if a robot can initially handle a basis task, its effective suitability for that task will never be zero. Hence, it will still try to participate in tasks that require those basis tasks for which it has a low effective suitability. This aids in the recovery of the robot from previous task failures that may or may not have been a result of actual malfunction.

III. Task Prioritization and Allocation

In general, this class of problems involving MR tasks can be formulated as a *set partitioning problem* (SPP) [1], [12], which is strongly NP-hard. The task-allocation problem, formulated as the SPP, is as follows [1].

Definition 3.1: Given a finite set of robots R and R_{fp}, a family of all feasible pairs of subsets of R (coalitions) and the associated task collectively termed as (coalition task) pairs, find a subset R_{part} of R_{fp} that maximizes the total utility derived from the partition, according to the utility function $u : R_{fp} \rightarrow \mathbb{R}$ for each (coalition task) pair, and such that $\bigcup R_{\text{part}} = R$ and $\bigcap R_{\text{part}} = \varnothing$.

If some robots are allowed to be idle, it is equivalent to assigning these robots to an arbitrarily defined idle task $\mathcal{T}_0$. The SPP is written as

$$\begin{aligned} \text{Maximize}: \quad & \sum_{i=1}^{n_{\text{comb}}} u(A_i)x_i \\ \text{subject to}: \quad & Ax = e \end{aligned} \tag{9}$$

where $x \in R_{\text{part}}$, $A \in \mathbb{R}^{(n_j+n_{ls})\times n_{\text{comb}}}$ is a matrix that contains all the elements of R_{fp} in its columns, $n_{\text{comb}} \leq n_{ls}\sum_{k=1}^{n_j} C_k^{n_j}$ is the cardinality of R_{fp}, u_i gives the utility derived from the coalition task pair represented by A_i (the ith column of A), and e is a vector containing ones. To reduce the complexity due to the nonlinear mapping from robot coalitions to a utility level for a task, we may reformulate the problem as a transportation problem (TP) by considering task suitabilities and assuming tasks gain from being allocated highly suited robots, i.e., robots whose intrinsic capabilities match the nature of the tasks as closely as possible. This is intuitive in real-life scenarios, *when robots do not have sufficient information* to estimate and compare the outcome of each possible team. The transformation thus converts the NP-hard problem to a simpler P-problem. The TP is stated as follows.

Definition 3.2: Given a set of supply points S, each capable of supplying a certain quantity of goods to elements in a set of demand points D, at a fixed cost/utility per unit (i.e., $f(s,d) \rightarrow c$, for $s \in S$, $d \in D$), the TP involves finding the amount of goods to be supplied from each supply point to satisfy the demand at each point in D, with minimal total cost (or maximum total utility).

Lemma 3.1: Assuming that it is always preferable for robots with high suitabilities for a task to serve that task, by using the notion of task suitability and restricting the number of robots required to service each task to exactly ϕ_i, the task-allocation problem in (9) can be reduced to the TP.

Proof: Since each coalition of robots can be assigned only one task (inclusive of $\mathcal{T}_0$), and each task to at most one coalition, the cardinality of $R_{\text{part}} \leq \min(n_{ls}+1, n_r)$. The partition R_{part} can be written as $R_{\text{part}} = \{(\Lambda_i, \mathcal{T}_i) | i = 0, \ldots, n_{ls}\}$, where Λ_i is the ith row of $\mathbf{\Lambda} \in \mathbb{Z}^{n_j \times (n_{ls}+1)}$, a (0,1) matrix. Intuitively, Λ_i represents a subset of robots from R (those corresponding to ones in Λ_i) that is paired with $\mathcal{T}_i$. The SPP can be reformulated as finding $n_{ls}+1$ vectors, each associated with a task, such that the total utility generated from all the coalition task pairs is maximized

$$\begin{aligned} \text{Maximize}: \quad & u(R_{\text{part}}) = \sum_{i=0}^{n_{ls}} u_i((\Lambda_i, \mathcal{T}_i)) \\ \text{subject to}: \quad & \sum_{k=0}^{n_{ls}} \mathbf{\Lambda}_{ik} = 1, \quad \text{for } i = 1, 2, \ldots, n_j. \end{aligned} \tag{10}$$

Consider the n_{sub} highest priority tasks in T_{ls}. This is reasonable when tasks are prioritized and the robots are unable to be assigned to all the tasks. Let $\mathbf{TSM}^{\text{sub}}(j)$ be a matrix containing the suitability levels of the robots with each of the n_{sub} tasks, inclusive of the idle task $\mathcal{T}_0$, with $\phi_0 = n_j - \sum_{k=1}^{n_{\text{sub}}} \phi_k$. Assuming that for a task requiring exactly ϕ_i robots, the task is better served by robots that have high suitabilities for it, the individual suitabilities can be linearly summed up,[4] and (10) can be rewritten as

$$\begin{aligned} \text{Maximize Suitability}: \quad & \sum_{k=0}^{n_{\text{sub}}+1} \sum_{i=1}^{n_j} \mathbf{\Lambda}_{ik} \mathbf{TSM}_{ik}^{\text{sub}}(j) \\ \text{subject to}: \quad & \sum_{k=0}^{n_{\text{sub}}+1} \mathbf{\Lambda}_{ik} = 1, \quad \text{for } i = 1, 2, \ldots, n_j \\ & \sum_{k_1=0}^{n_j} \mathbf{\Lambda}_{k_1 i_1} = \phi_{i_1}, \quad \text{for } i_1 = 1, 2, \ldots, n_{atv}+1. \end{aligned} \tag{11}$$

The above formulation may be viewed as a TP [23] by letting tasks and robots correspond to the demand and supply nodes, respectively, of a typical TP. Each supply node provides exactly one robot, while each task has a demand for ϕ_i robots. The revenue generated by satisfying a task's demand with a robot is given by the robot's suitability for the task, and the objective of the TP considered here will be to maximize suitability. ■

Since ϕ_i remains fixed during the allocation process, the robots will face a TP each time (re)allocation occurs. Note that in the normal sense of utility, a robot team's utility for a task may not be a combination of the robots' individual utilities. However, the use of task suitability, together with the TP, helps determine a team (of the required size) with robots that are highly suited for each task, such that the team has a high overall suitability (and chance of success). Furthermore, although each task may be viewed as a collection of subtasks, these need not be independent.

Remark 3.1: If COBOS is implemented using standard definitions of utilities, the reduction of the SPP to the TP will only hold for the set of tasks that can be decomposed into a set of subtasks that can be executed independently.

In this paper, optimality is defined as follows.

Definition 3.3 (Optimality): For a given set of robots and tasks, and each robot having a suitability level for each task, a task allocation is optimal when the number of robots servicing each task is the smallest (i.e., robot teams of sizes less than the current ϕ_i is unable to perform the task), and the total suitability value is maximum.

[4]The problem of forming infeasible coalitions is avoided by assigning $-\infty$ (or a very large negative value) suitability values, instead of zero, to tasks a robot cannot handle.

TABLE II
COMPARISON BETWEEN DIFFERENT TASK-ALLOCATION ARCHITECTURES. FOR SIMPLICITY, $r = n_r$ AND $n = n_{ls}$. THE FIGURES PERTAINING TO ALLIANCE, BLE, M+, AND DYNAMIC ROLE ASSIGNMENT ARE BASED ON THE WORK PRESENTED IN [1]

Architecture	Operation Domain			Communication	Computation	Quality
	General	Task Specs.	n_{sn}			
ALLIANCE	ST-SR-IA/TA	fully known	$=1$	$O(r)$	$O(rn)$	$>$ 2-Competitive
BLE	ST-SR-IA	fully known	$=1$	$O(rn)$	$O(rn)$	2-Competitive
COBOS (and solving TP)	ST-MR-IA	limited knowledge	≥ 1	$O(rn)$	$O(r^3)$	Optimal ($\|N\|=1$) $>$ 2-Competitive ($\|N\|>1$)
COBOS (and using Greedy Alloc.)	ST-MR-IA	limited knowledge	≥ 1	$O(rn)$	$O(rn)$	2-Competitive
Dynamic Role Assignment	ST-SR-IA	fully known	$=1$	$O(r)$	$O(r)$: Auctioneer $O(1)$: Bidder	$\geq$ 3-Competitive
M+	ST-SR-IA	fully known	$=1$	$O(rn)$	$O(rn)$	2-Competitive

Theorem 3.2: Optimal allocation of robots to the n_{atv} feasible and most important tasks in the system is achieved with COBOS by solving a TP, under the condition that all tasks lie within the main broadcast region N_0.

Proof: By construction, the size of coalitions required for each task is constrained to be ϕ_i. When all the tasks are within N_0, by Algorithm 1, robots always consider all the available tasks, and $\mathbf{TSM}^{-}(j)$ will contain the n_{atv} tasks with the highest priorities. Hence, by *Lemma 3.1*, each robot r_j will be solving a TP associated with $\mathbf{TSM}^{-}(j)$, and in the form of (11). Such optimization problems may be easily solved, e.g., using the Hungarian Algorithm [of $O(n_j^3)$] to produce an allocation with maximum suitability. However, since COBOS adjusts the values of ϕ_i, the convergence of a present allocation to one, where tasks with the highest priorities get the needed number of robots, depends on the convergence of the initial TP to a stable TP, i.e., ϕ_i stops changing for the tasks in $\mathbf{TSM}^{-}(j)$. As described in Section II-D, each task needs a default of one robot. Let $\mathcal{T}_{\text{do}} \subseteq \mathcal{T}$ be the set of n_{atv} tasks under consideration. As the operation progresses, if a task $\mathcal{T}_i \in \mathcal{T}_{\text{do}}$ is inadequately performed, ϕ_i will be incremented, which correspondingly influences the size of $\mathcal{T}_{\text{do}}$ and n_{atv}.

Consider $n_{ls} \leq n_r$, where n_r is the total number of robots. In the beginning, $n_{atv} = n_{ls}$, since each task is assumed to require only one robot, and there will be $n_r - n_{ls}$ extra robots. If the stable TP is not reached (i.e., there are not enough robots servicing at least one of the tasks in $\mathcal{T}_{\text{do}}$), at least one task in $\mathcal{T}_{\text{do}}$ will increase their demand for robots, and a maximum of $n_r - n_{ls}$ adaptations of ϕ will occur before the pool of extra robots are fully deployed. If the stable TP is still not reached, an increase in robot demand by any task in $\mathcal{T}_{\text{do}}$ would result in excess demand. COBOS hence reduces n_{atv} (and the size of $\mathcal{T}_{\text{do}}$) to include only the high-priority tasks that can be serviced. The worst case resulting from the reduction of n_{atv} would be to free up $n_r - (n_{ls} - 1)$ robots, originally servicing the n_{ls}th task. These robots will be deployed, in at most $n_r - (n_{ls} - 1)$ adaptations, to satisfy any increased demands. This continues until a stable TP, or the terminating case (where the highest priority task in $\mathcal{T}_{\text{do}}$ requires n_r robots to perform) is reached. Considering the worst-case scenario to reach the terminating case, at most $\sum_{k=0}^{n_{ls}-1}(n_r - (n_{ls} - k))$ adaptations of ϕ are required. The analysis is similar for cases where $n_{ls} > n_r$, except with $n_{atv} = n_r$ at the beginning, since COBOS will only consider the n_r highest priority tasks. The upper bound on the number of adaptations of ϕ will be $\sum_{k=1}^{n_r-1} k$.

Since the time between operations mainly depends on the backoff and expected task-completion times (which are finite), together with a finite n_r and n_{ls}, the total time taken for the TP to converge to a stable TP is finite. ■

A robot selects the task that it should perform by solving the TP associated with $\mathbf{TSM}^{-}(j)$. For the whole system, because of eventually identical TSMs between robots within the same subnetwork (Section II-B), each robot will arrive at the same allocation decision. Hence, the combined actions of the robots result in an optimal allocation for the whole system.

IV. ANALYSIS AND COMPARISONS

The four architectures used for comparison purposes with COBOS are: 1) ALLIANCE [2]; 2) BLE [4]; 3) M+ [6]; and 4) dynamic role assignment [9]. Comparisons are made regarding: 1) the domain of operation, and based on the work in [1]; 2) communication requirements; 3) computation requirements; and 4) quality of solution. A summary of the comparisons is shown in Table II.[5]

A. Domain of Operation

1) Multirobot Tasks and Uncertain Task Requirements: As discussed earlier, COBOS operates in ST-MR-IA domains, and task-specific information is not available for *a priori* division of tasks into SR tasks. In contrast, most of the other architectures had been designed for ST-SR-IA/TA systems. Applied to ST-SR-IA systems, COBOS will still be able to generate appropriate allocations, except that the recruitment phase will always be off.

2) Isolated Pockets of Communication Networks:

Proposition 4.1: COBOS is able to guarantee feasible allocation of robots to tasks in the presence of disjoint broadcast networks, given that all the robots are initially located within the main broadcast network N_0.

Proof: The condition that all the robots start from within N_0 allows them to receive the initial set of tasks. Let $N_{-0} = \bigcup_{i=1}^{n_{\text{sn}}} N_i$. Assume that there is only one task related to each subnetwork in N_{-0}. The recruitment process is the same for subnetworks with more tasks, except that the recruiting robot(s) does not need to travel to another subnetwork if it is recruiting from tasks within its own subnetwork. In the beginning, one robot will be allocated to each task (subnetwork). If the robot is unable to complete the task, it enters the recruitment phase and visits each task in visitinglist$_j(\cdot)$ until it recruits enough robots. Since tasks in visitinglist$_j(\mathcal{T}_k)$ are of lower priority than $\mathcal{T}_k$, a

[5]Analysis of ALLIANCE, BLE, M+, and dynamic role assignment based on communication, computation, and solution quality is presented in [1].

higher priority task requiring assistance will obtain help from those of lower priorities. Therefore, a feasible allocation will always result for the current subset of tasks that the team has decided to service. Recruiting tasks will get the number of robots they require, and tasks that are currently not considered will be serviced when robots become available. ■

Systems with disjoint broadcast networks have not been considered in current task-allocation schemes. In addition, for many schemes, a robot is deemed to be malfunctioning when the task it is handling shows no progress. This is, in general, untrue for systems with disjoint subnetworks. Furthermore, bidding mechanisms, where the auctioneer is located at the site of the task involved, are not equipped to enlist the help of robots in other subnetworks. In cases where there is only one broadcast network, *and* sufficient information to allow users to specify the number of robots required per task, and determine the effectiveness of coalitions, the performance of COBOS is similar to that of other work in the ST-SR-IA domain.

B. Communication Complexity

Communication complexity reflects the number of messages a robot broadcasts per execution cycle of the algorithm. For COBOS, a robot r_j broadcasts its suitabilities for each task, i.e., n_{ls} messages. Hence, communication overhead is $O(n_j n_{ls})$. In addition, each robot also broadcasts its chosen task and the task's attributes (recruitFlag, ϕ_i, $TTry(i)$, and $\mathbf{T}_i$). This contributes an additional $5n_j$ messages, and the total communication is $O(n_j(n_{ls}+5)) \approx O(n_j n_{ls})$. In comparison, COBOS has a higher dependence on communications, although the order of communication complexity is approximately the same as BLE and M+, as shown in Table II.

C. Computation Complexity

The computational requirement of an algorithm is measured in terms of the frequency that one or more resource-expensive operation(s) is executed by each robot. Each robot using COBOS mainly uses linear programming approaches (of complexity $O(n_j^3)$ [23]) to solve a TP. As shown in Table II, computational complexity is $O(rn)$ for most other cases except for dynamic role assignment ($O(r)$ for the auctioneer), and COBOS requires more processing power. However, by sacrificing optimality, a greedy mechanism that considers all tasks in a subnetwork (similar to the BLE), may also be used in place of solving the TP for COBOS' allocation process, resulting in a computational complexity of $O(rn)$. This does not influence the actual operation of COBOS in handling subnetworks and uncertain task specifications.

D. Quality of Solutions

From *Theorem 3.2*, COBOS can produce instantaneous optimal allocation. For the comparison of solution quality (except for dynamic role assignment that sorts tasks according to arrival times, and does not handle task priorities), we account for task prioritization and consider the performance of allocating tasks to the first n_{atv} tasks of T_{ls}. If task suitability is used for computing bids, the main difference between the four schemes and COBOS would be that for COBOS, each robot compiles all the information (e.g., bids) and makes a decision based on all the information, instead of handling the information task by task.

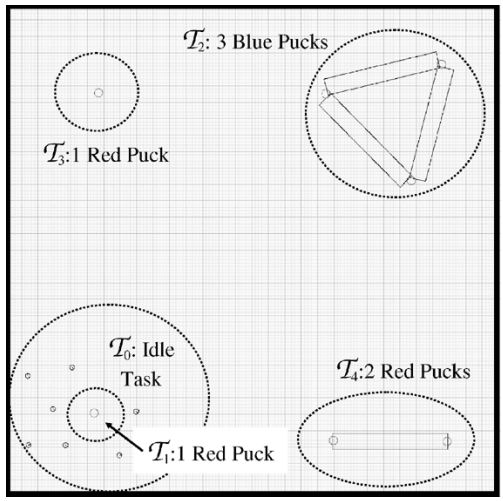


Fig. 4. Closed room (30 m×30 m) with multiple communication networks.

The allocation schemes considered here employ some form of the greedy mechanism, which is well known to be 2-competitive (i.e., solution is at least 1/2 as good as optimal). On the other hand, from *Theorem 3.2*, COBOS allows optimal allocation in the presence of one connected broadcast network. If a greedy mechanism is used for COBOS, the performance will be similar to that of other algorithms. For disjoint broadcast networks, we have the following.

Proposition 4.2: COBOS results in locally (in the sense of subnetworks) optimal allocation, when physical conditions result in disjoint broadcast subnetworks, and the TP is solved.

Proof: Consider a subnetwork $N_{i_{\text{sn}}} \in N_{-0}$ associated with the set of tasks $T_{i_1} \subseteq T_{ls}$ (for $i_1 = 1, \ldots, n_{i_{\text{sn}}}$), such that $\mathcal{L}_{i_1} \in N_{i_{\text{sn}}}$. By construction, each robot considers only the tasks within its subnetwork, plus any recruiting tasks (lines 1–6 of Algorithm 1). Recruiting tasks are transient, i.e., if robots are allocated to service the recruiting tasks, they will ultimately leave the subnetwork. Hence, the robots in $N_{i_{\text{sn}}}$ eventually consider only the tasks in T_{i_1}. Any allocation that occurs within this subnetwork will thus be optimal only locally. ■

The locally optimal allocation for the case in *Proposition 4.2* may be seen as a form of greedy assignment, since optimization is performed for each subgroup that arises due to the subnetworks, which "greedily" selects the robots it wants. Since optimization is performed within each subgroup, the solution will be at least 2-competitive.

V. Simulation Experiments

Realistic simulations were carried out using Player and Stage [24]. The simulated environment is shown in Fig. 4. The team consists of seven heterogeneous robots, and their capabilities and the suitability levels with each basis task is given in Table III. The *find-and-attach* basis tasks are chosen for convenience and simplicity and may easily be replaced by others, e.g., transportation between two locations, for more elaborate missions. The following five tasks are presented to the robot team.

1) $\mathcal{T}_0$: Idle task. $\mathcal{T}_0 = [1 \quad 0 \quad 0] \odot [\{\mathcal{L}_0\} \quad \times \quad \times]^T$

TABLE III
ROBOT CAPABILITIES AND SUITABILITY LEVELS FOR BASIS TASKS AND TASKS. B: BLUE, R: RED. TASKS ARE PRIORITIZED USING THEIR SUBSCRIPT. (NOTE: INFORMATION REGARDING TASK 4 IS PRESENTED HERE FOR CONVENIENCE. IT IS, IN FACT, INTRODUCED ONLY AT RUNTIME AS A NEW TASK)

	Equipment		**Basis Tasks** (Initial Suitability)			**Tasks** (Initial Suitability)				
	Perceived	**Actual**	$Goto(\cdot)$	$blue\_find\_attach(\cdot)$	$red\_find\_attach(\cdot)$	$\mathcal{T}_0$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$
R1	B,R	B	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R2	B,R	R	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R3	B,R	B	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R4	B,R	R	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R5	B,R	B	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R6	B,R	R	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9
R7	B,R	B,R	0.9	0.9	0.9	-	0.9	0.9	0.9	0.9

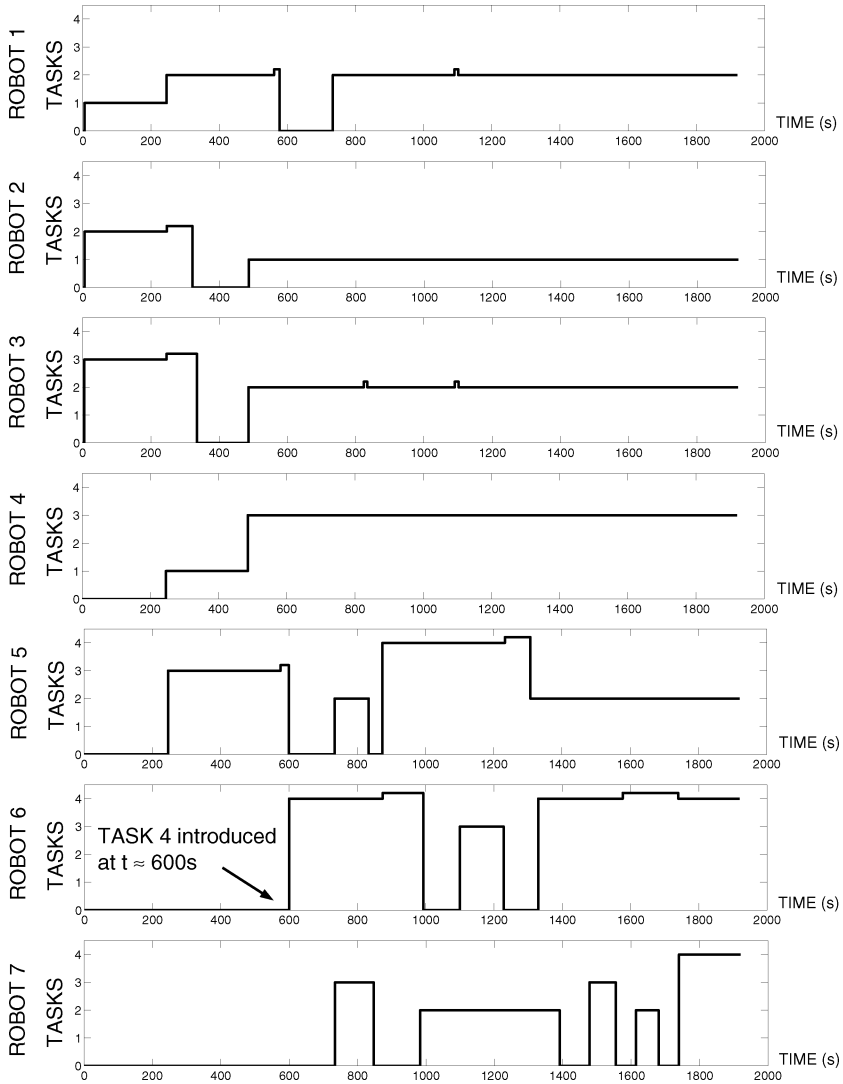
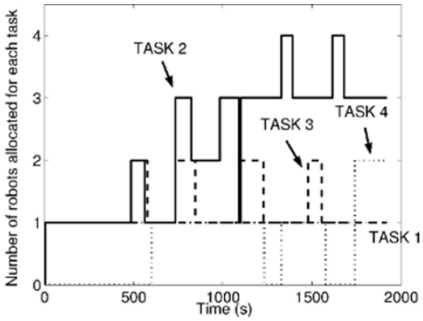


Fig. 5. Activity charts of the robots in the presence of multiple subnetworks. The small kinks in the graphs, characterized by slightly raised lines, indicate that the robot is in recruitment phase for the associated task.

2) $\mathcal{T}_i$ (for $i = 1, 3, 4$): To locate and attach to all the **red** pucks at $\mathcal{L}_i$. The task is completed or adequately performed only when all the pucks are simultaneously in contact with robots.
3) $\mathcal{T}_i = [1 \quad 0 \quad 1] \odot [\mathcal{L}_0 \quad \times \quad \mathcal{L}_i]^T$.
4) $\mathcal{T}_2$: Similar to tasks 1, 3, and 4, except that the pucks are **blue**.
5) $\mathcal{T}_2 = [1 \quad 1 \quad 0] \odot [\mathcal{L}_0 \quad \mathcal{L}_i \quad \times]^T$.

More than 30 simulation runs were performed, with different numbers of robots required for each task. Since only the nature of the tasks are defined, task specifications remain unchanged. This is consistent with the fact that users are usually unable to provide all the specifics of the task. The number of colored pucks at each location is unknown to the robots. In addition, some robots can only locate pucks of one color, although they are initially assumed to be fully capable of detecting both colors. Furthermore, the broadcasted messages of a robot can only reach others within 10 m from it.

Remark 5.1: For experiments with physical robots, when communication imperfections exist *within* a subnetwork, a



(a) Number of robots serving each task. Robots in recruitment phase are not included.

	b1	b2
R1	0.9	0.6
R2	0.6	0.9
R3	0.9	0.6
R4	0.9	0.9
R5	0.9	0.36
R6	0.9	0.9
R7	0.9	0.9

(b) Suitability Levels for basis tasks, $b1$: $blue\_find\_attach(\cdot)$, $b2$: $red\_find\_attach(\cdot)$.

Fig. 6. Robot allocation per task, and robot suitability for basis tasks at the end of mission, for domain with multiple subnetworks.

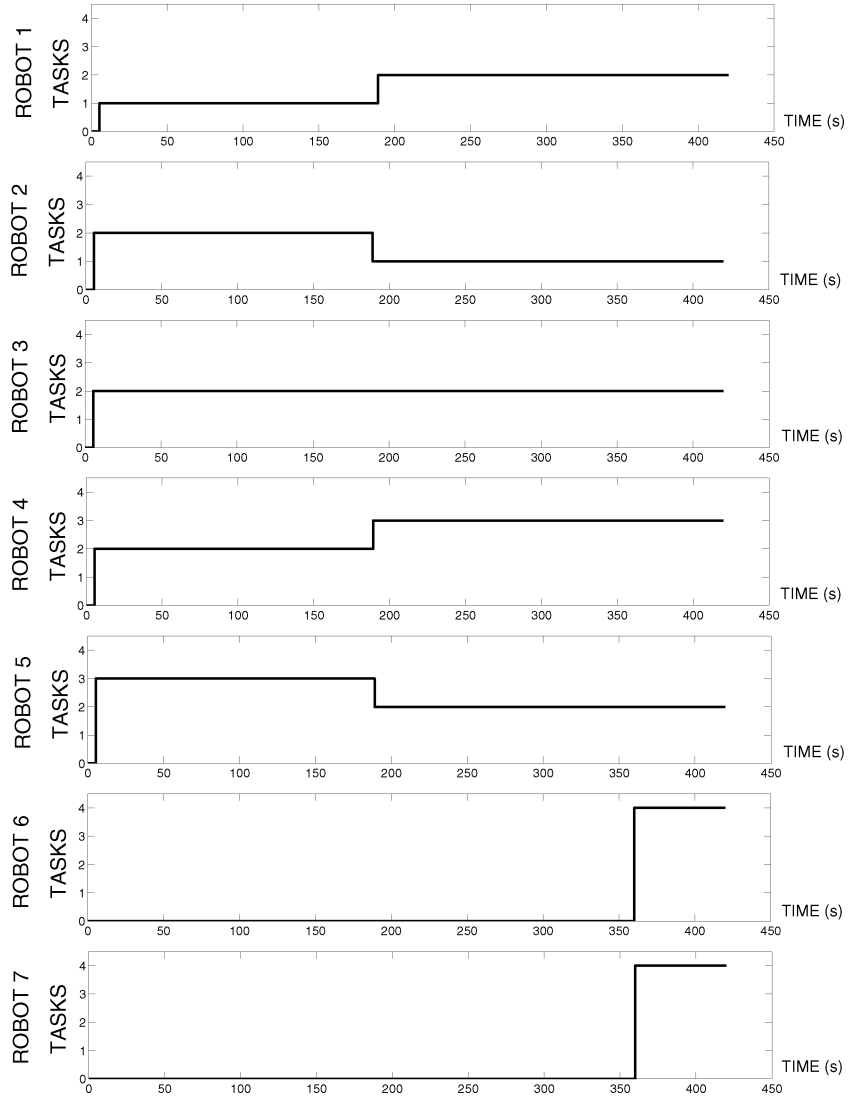


Fig. 7. Activity charts of the robots with one connected network and no uncertain task specifications.

forced short information-synchronization period may be necessary to delay the robots' allocation process, to ensure that they have all the required information regarding other robots (within the same subnetwork) before making decisions.

The expected maximum completion times of all tasks are set to 250 s, and ϵ (8) is set to 0.5. The activity charts of the robot team, and the number of robots allocated per task over time, are shown in Figs. 5 and 6(a), respectively. After the initial allocation, r_2 and r_3 moved out of N_0. When the tasks cannot be completed, the robots enter the recruitment phase. At the same time, $\mathcal{T}_2$ and $\mathcal{T}_3$ lapse in N_0, triggering a reallocation in N_0, where r_1 and r_5 take up $\mathcal{T}_2$ and $\mathcal{T}_3$, respectively. r_1, unable to complete $\mathcal{T}_2$, enters the recruitment phase, and eventually forms a team of two with r_3 at $t \approx 750$ s. r_5 being unable to detect red

pucks gets replaced by r_4 at $t \approx 600$ s. Task 4 was introduced at $t \approx 600$ s. Only the robots in N_0 are aware of the task, and r_6 was sent. Since $\mathcal{T}_4$ is of the lowest priority, extra robots are always sent first to the subnetworks with higher priority tasks when these tasks are observed (from N_0) to have lapsed. This accounts for the relatively long period that r_6 remains in recruitment mode ($1600 \text{ s} \leq t \leq 1700$ s) before r_7 starts $\mathcal{T}_4$. Despite disjoint subnetworks, the robots are still able to reach the suitable allocation for all the tasks by $t \approx 1800$ s. The suitabilities of each robot to the basis tasks after allocation is complete is shown in Fig. 6(b).

When the *number of robots required per task is known*, in addition to having a single broadcast network, the robot allocations using COBOS augmented with the greedy mechanism are shown in Fig. 7. The tasks are considered in order of priority, and robots are assigned according to suitability. Reallocation is performed when a task lapses. Although there are fewer fluctuations in the allocations, the number of robots per task is assumed to be known, in the absence of which, a suitable distribution cannot be achieved without COBOS.

VI. Conclusions and Future Work

COBOS, a fault-tolerant task-allocation scheme for ST-MR-IA domains with restrictive communication, has been presented. It can accommodate uncertainties in task specifications due to incomplete knowledge. This eases the requirements made on the sensing capabilities of robots and high-level task-decomposition algorithms. Flexibility and fault detection are further increased through COBOS' adaptive capabilities. The scheme can potentially be used in marsupial teams (see chapter by Murphy in [18]). Simulations verify the effectiveness of the proposed scheme.

Acknowledgment

The authors would like to thank the Editor, S. Hutchinson, the Associate Editor, G. Sukhatme, and the anonymous reviewers for their valuable comments and suggestions for improving earlier versions of this paper.

References

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[2] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[3] ——, "L-ALLIANCE: Task-oriented multirobot learning in behavior-based systems," *Adv. Robot.*, vol. 11, no. 4, pp. 305–322, 1997.

[4] B. B. Werger and M. J. Matarić, "Broadcast of local eligibility for multitarget observation," in *Distributed Autonomous Robotic Systems*, L. E. Parker, G. Bekey, and J. Barhen, Eds. New York: Springer, 2001, vol. 4, pp. 347–356.

[5] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 769–780, Oct. 2002.

[6] S. C. Botelho and R. Alami, "M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proc. Int. Conf. Robot. Autom.*, May 1999, pp. 1234–1239.

[7] R. Zlot and A. Stentz, "Complex task allocation for multiple robots," in *Proc. IEEE Int. Conf. Robot Autom.*, Apr. 2005, pp. 1527–1534.

[8] N. Kalra, T. Stentz, and D. Fergusen, "Hoplites: A market framework for complex tight coordination in multi-agent teams," Carnegie Mellon Univ., Robot. Inst., Pittsburgh, PA, Rep. CMU-RI-TR-04-41, 2004.

[9] L. Chaimowicz, M. F. M. Campos, and V. Kumar, "Dynamic role assignment for cooperative robots," in *Proc. Int. Conf. Robot. Autom.*, May 2002, pp. 293–298.

[10] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[11] R. Emery, K. Sikorski, and T. Balch, "Protocols for collaboration, coordination and dynamic role assignment in a robot team," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, pp. 3008–3015.

[12] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1-2, pp. 165–200, May 1998.

[13] M. B. Dias, M. Zinck, R. Zlot, and A. Stentz, "Robust multirobot coordination in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2004, pp. 3435–3442.

[14] C. Fua, S. S. Ge, and K. W. Lim, "BOAs: Back-off adaptive scheme for task allocation with fault tolerance and uncertainty management," in *Proc. Int. Symp. Intell. Control*, Sep. 2004, pp. 162–167.

[15] L. E. Parker, "The effect of heterogeneity in teams of 100+ mobile robots," in *Multi-Robot Systems*. Norwell, MA: Kluwer, 2003, vol. II, From Swarms to Intelligent Automata, pp. 205–215.

[16] S. S. Ge and C. Fua, "Queues and artificial potential trenches for multirobot formations," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 646–656, Aug. 2005.

[17] ——, "Complete multi-robot coverage of unknown environments with minimum repeated coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 727–732.

[18] T. Balch and L. E. Parker, Eds., *Robot Teams: From Diversity to Polymorphism*. Natick, MA: A. K. Peters, 2001.

[19] D. Gauthier, P. Freedman, G. Carayannis, and A. S. Malowany, "Interprocess communication for distributed robotics," *IEEE J. Robot. Autom.*, vol. RA-3, no. 6, pp. 493–504, Dec. 1987.

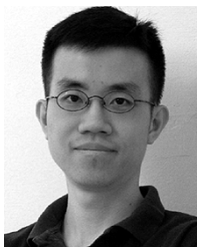
[20] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1157–1162.

[21] F. Tang and L. E. Parker, "ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1513–1520.

[22] A. Stroupe, R. Ravichandran, and T. Balch, "Value-based action selection for exploration and dynamic target observation with robot teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2004, pp. 4190–4197.

[23] G. B. Dantzig and M. N. Thapa, *Linear Programming*. New York: Springer, 1997, vol. 1.

[24] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. Adv. Robot.*, Jun.–Jul. 2003, pp. 317–323.



Cheng-Heng Fua (S'03) received the B.Eng. degree in electrical and computer engineering from the National University of Singapore (NUS), Singapore, in 2003, where he is currently working toward the Ph.D. degree with the NUS Graduate School for Integrative Sciences and Engineering.

His research interests are in the area of autonomous multirobot collaboration and planning.

Mr. Fua is the recipient of the A*Star Graduate Scholarship, awarded by the Agency for Science, Technology, and Research (A*Star), Singapore.

Shuzhi Sam Ge (S'90–M'92–SM'00) received the B.Sc. degree from Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, in 1986, and the Ph.D. degree and the Diploma of Imperial College (DIC) from the Imperial College of Science, Technology, and Medicine, University of London, London, U.K., in 1993.

He has been with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, since 1993, where he is currently a Professor. He has authored and coauthored over 200 international journal and conference papers, three monographs, and coinvented three patents. He has been serving as the (Associate) Editor of several leading international journals. He has also been serving as a technical consultant for the local industry. His current research interests are control of nonlinear systems, neural/fuzzy systems, robotics, hybrid systems, sensor fusion, and system development.